BACCALAURÉAT

SESSION 2023

Épreuve de l'enseignement de spécialité

NUMÉRIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°04

DURÉE DE L'ÉPREUVE : 1 heure

Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (4 points)

Écrire une fonction `a_doublon` qui prend en paramètre une liste **triée** de nombres et renvoie `True` si la liste contient au moins deux nombres identiques, `False` sinon.

Par exemple :

```
>>> a_doublon([])
False

>>> a_doublon([1])
False

>>> a_doublon([1, 2, 4, 6, 6])
True

>>> a_doublon([2, 5, 7, 7, 7, 9])
True

>>> a_doublon([0, 2, 3])
False
```

EXERCICE 2 (4 points)

On souhaite générer des grilles du jeu de démineur à partir de la position des bombes à placer.

On se limite à la génération de grilles carrées de taille $n \times n$ où n est le nombre de bombes du jeu.

Dans le jeu du démineur, chaque case de la grille contient soit une bombe, soit une valeur qui correspond aux nombres de bombes situées dans le voisinage direct de la case (au-dessus, en dessous, à droite, à gauche ou en diagonal : chaque case a donc 8 voisins si elle n'est pas située au bord de la grille).

Voici un exemple de grille 5x5 de démineur dans laquelle la bombe est représentée par une étoile :

1	1	1	0	0
1	*	1	1	1
2	2	3	2	*
1	*	2	*	3
1	1	2	2	*

On utilise une liste de listes pour représenter la grille et on choisit de coder une bombe par la valeur -1.

L'exemple ci-contre sera donc codé par la liste :

```
[[1,  1,  1,  0,  0],
 [1, -1,  1,  1,  1],
 [2,  2,  3,  2, -1],
 [1, -1,  2, -1,  3],
 [1,  1,  2,  2, -1]]
```

Compléter le code suivant afin de générer des grilles de démineur, on pourra vérifier que l'instruction `génère_grille([(1, 1), (2, 4), (3, 1), (3, 3), (4, 4)])` produit bien la liste donnée en exemple.

```
def voisinage(n, ligne, colonne):
    """ Renvoie la liste des coordonnées des voisins de la case
       (ligne, colonne) en gérant les cases sur les bords. """
    voisins = []
    for l in range(max(0,ligne-1), min(n, ligne+2)):
        for c in range(max(0, colonne-1), min(n, colonne+2)):
            if (l, c) != (ligne, colonne):
                voisins.append((l,c))
    return voisins

def incremente_voisins(grille, ligne, colonne):
    """ Incrémente de 1 toutes les cases voisines d'une bombe."""
    voisins = ...
    for l, c in voisins:
        if grille[l][c] != ...: # si ce n'est pas une bombe
            ...                 # on ajoute 1 à sa valeur

def genere_grille(bombes):
    """ Renvoie une grille de démineur de taille nxn où n est
        le nombre de bombes, en plaçant les bombes à l'aide de
        la liste bombes de coordonnées (tuples) passée en
        paramètre. """
    n = len(bombes)
    # Initialisation d'une grille nxn remplie de 0
    grille = [[0 for colonne in range(n)] for ligne in range(n)]
    # Place les bombes et calcule les valeurs des autres cases
    for ligne, colonne in bombes:
        grille[ligne][colonne] = ...  # place la bombe
        ...                           # incrémente ses voisins
    return grille
```